Date=07/08/2020
Lecture By=Shubham Joshi
Subject ⇒Stacks Problems Solving

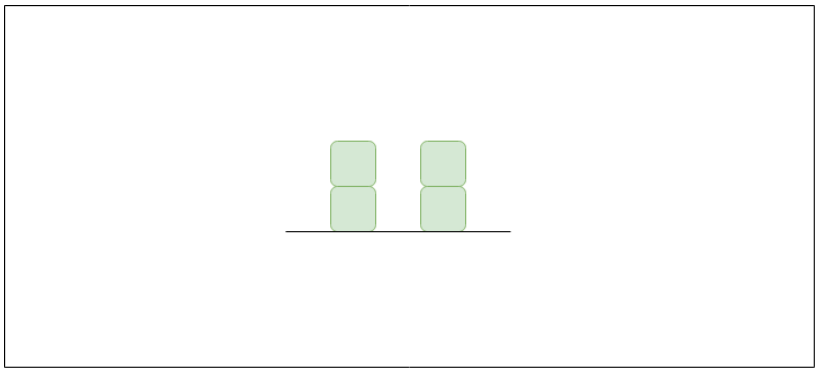
IN PREVIOUS LECTURE **(QUICK RECAP) Date-06/08/2020**	**In Today's Lecture (Overview)**
Stacks in python Question=1 Implement stack Using LINKED LIST Question=2Given an array Print Next Greater Element MCQs Questions For Self Practice	Questions Regarding Stacks Question=Trapping Rain Water MCQ's Questions For Self Practice // CC AND Assignment For The Day

Question=Trapping Rain Water

Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it is able to trap after raining.

Example1

```
Input: arr[]   = {2, 0, 2}
Output: 2
Explanation:
The structure is like below
```



We can trap 2 units of water in the middle gap.

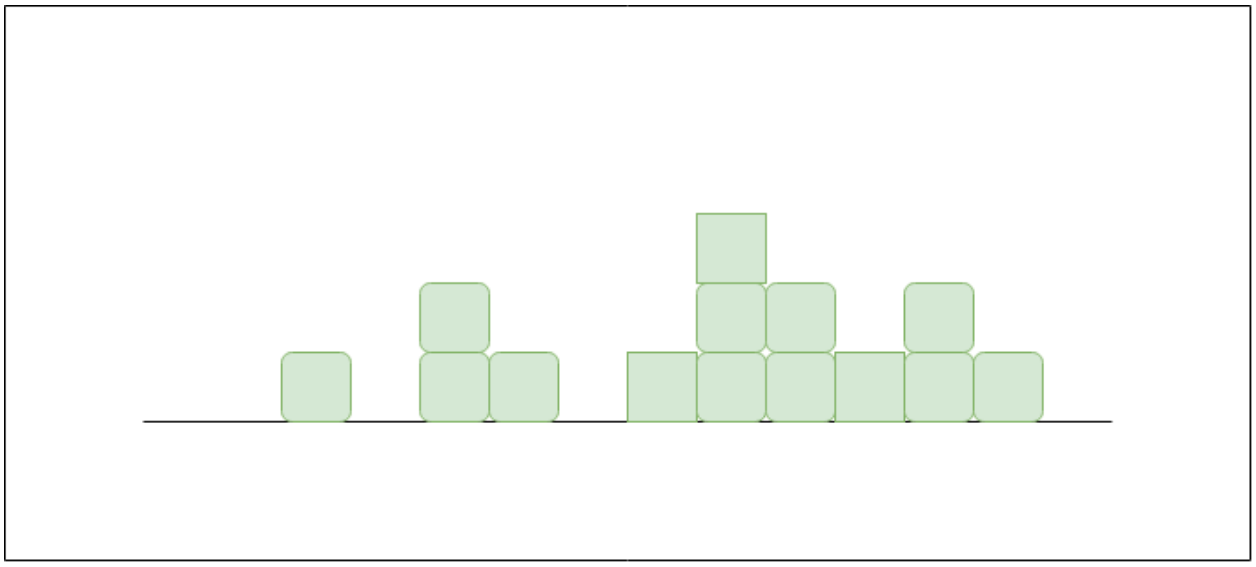
Example2

```
Input: arr[] = [0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1]
Output: 6
```

Explanation:
We can trap "3 units" of water between 3 and 2,

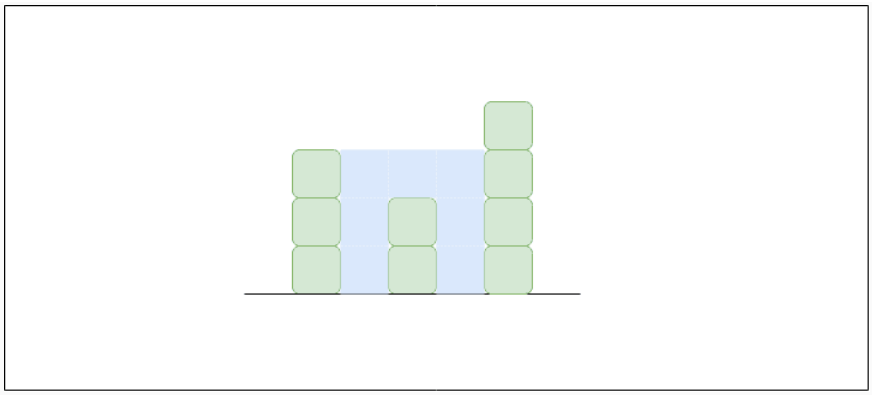
"1 unit" on top of bar 2 and "3 units" between 2

and 4. See below diagram also.



Example3

Consider the array {3, 0, 0, 2, 0, 4}, three units of water can be stored three indexes 1 and 2, and one unit of water at index 3, and three units of water at index 4.
For Array[] = {3, 0, 2, 0, 4}
Water stored = 0 + 3 + 1 + 3 + 0 = 7



Explanation

Approach: The idea is to traverse every array element and find the highest bars on left and right sides. Take the smaller of two heights. The difference between the smaller height and height of the current element is the amount of water that can be stored in this array element.

Algorithm:

1. Traverse the array from start to end.
2. For every element, traverse the array from start to that index and find the maximum height *(a)* and traverse the array from the current index to end and find the maximum height *(b)*.
3. The amount of water that will be stored in this column is *min(a,b) – array[i]*, add this value to total amount of water stored
4. Print the total amount of water stored.

Code

```
# water that can be stored
def maxWater(arr, n):
    size = n - 1

    # Let the first element be stored as
    # previous, we shall loop from index 1
    prev = arr[0]
```

```
    # To store previous wall's index
    prev_index = 0
    water = 0

    # To store the water until a larger wall
    # is found, if there are no larger walls
    # then delete temp value from water
    temp = 0
    for i in range(1, size + 1):

        # If the current wall is taller than
        # the previous wall then make current
        # wall as the previous wall and its
        # index as previous wall's index
        # for the subsequent loops
        if (arr[i] >= prev):
            prev = arr[i]
            prev_index = i

            # Because larger or same height wall is found
            temp = 0
        else:

            # Since current wall is shorter than
            # the previous, we subtract previous
            # wall's height from the current wall's
            # height and add it to the water
            water += prev - arr[i]

            # Store the same value in temp as well
            # If we dont find any larger wall then
            # we will subtract temp from water
            temp += prev - arr[i]

    # If the last wall was larger than or equal
    # to the previous wall then prev_index would
    # be equal to size of the array (last element)
```

```
    # If we didn't find a wall greater than or equal
    # to the previous wall from the left then
    # prev_index must be less than the index
    # of the last element
    if (prev_index < size):

        # Temp would've stored the water collected
        # from previous largest wall till the end
        # of array if no larger wall was found then
        # it has excess water and remove that
        # from 'water' var
        water -= temp

        # We start from the end of the array, so previous
        # should be assigned to the last element
        prev = arr[size]

        # Loop from the end of array up to the 'previous index'
        # which would contain the "largest wall from the left"
        for i in range(size, prev_index - 1, -1):

            # Right end wall will be definitely smaller
            # than the 'previous index' wall
            if (arr[i] >= prev):
                prev = arr[i]
            else:
                water += prev - arr[i]

    # Return the maximum water
    return water

# Driver code
arr = [0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1]
n = len(arr)
print(maxWater(arr, n))
```

Output
6

MCQ’s

1.**where do you think stack is used in industry ?**

A.for doing undo / redo

B.replacement for stack

C.for doing complex computation

2.**What is the time and space complexity for min stack ?**

A.O(1) time O(n) space

B.O(n) time O(1) space

C.O(1) time O(1) space

3.**What is the time complexity of insertion sort ?**

A.O(N)

B.O(N2)

C.O(NLOGN)

Questions For Self Practice // CC AND Assignment For The Day

1.Implement Stack Using Linked List

2. https://leetcode.com/problems/min-stack/